

# Functionless batch processing with EB

1. About me
2. Situation
3. What we did
4. What we learnt

# About me

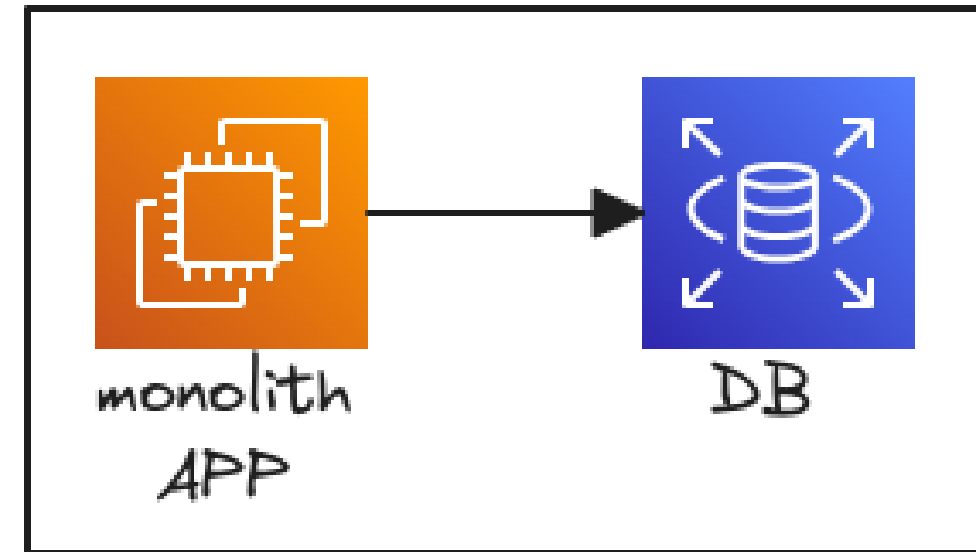
- I am Carlos Angulo
- I work at [Ohpen](#) as a Platform Engineer
- I have more than 8 years of experience
- You can find me on [LinkedIn](#) or [GitHub](#)



# Situation

We have a legacy monolith app running in EC2 + RDS

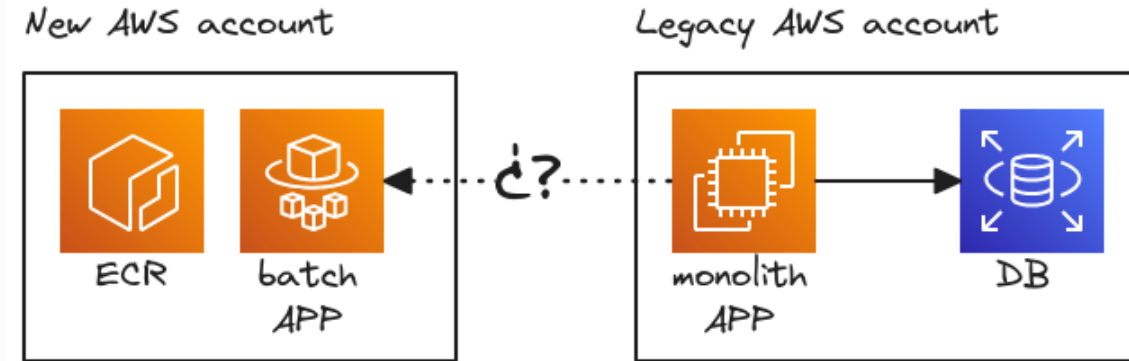
Legacy AWS account



# Dockerization

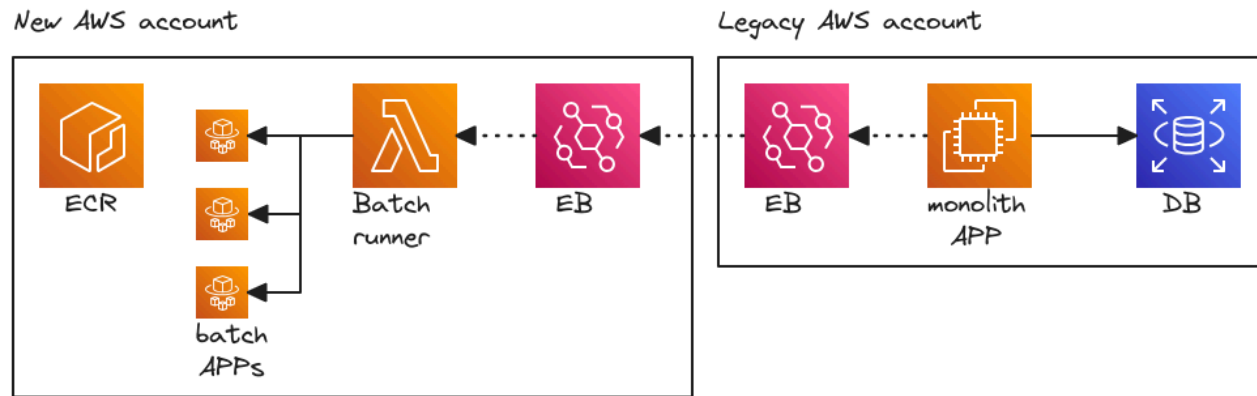
We dockerized some batch apps in a new account

*How would you trigger those apps from the monolith?* 🤔



# What we did


We decide to trigger  
batches based on  
events



```
1 public async Task RunBatch(EventData detail, string traceId)
2 {
3     var ecsConfig = await _configProvider.GetConfig(detail, traceId);
4     var request = new RunTaskRequest
5     {
6         Cluster = ecsConfig.ClusterName,
7         TaskDefinition = ecsConfig.TaskDefinition,
8         NetworkConfiguration = new NetworkConfiguration
9         {
10             ... // omitted for brevity
11         },
12         Overrides = new TaskOverride
13         {
14             ContainerOverrides = new List<ContainerOverride>
15             {
16                 new()
17                 {
18                     Name = ecsConfig.TaskDefinition,
19                     Command = ecsConfig.CommandParams,
20                     Environment = ecsConfig.EnvironmentVariables
21                         .Select(source => new EcsKeyValuePair() { Name = source.Key, Value = source.Value })
22                         .ToList()
23                 }
24             }
25         }
26     };
27
28     var response = await _ecsClient.RunTaskAsync(request);
29 }
```



```
1 public async Task<EcsBatchConfig> GetConfig(EventData detail, string traceId)
2 {
3     var batchInfo = _supportedBatches.FirstOrDefault(b => b.BatchTypeId == detail.BatchTypeId);
4     if (batchInfo == null)
5         throw new NotImplementedException("Unsupported batch type");
6
7     var conf = _batchConfigurationProvider.Get(batchInfo.Id)
8     var config = new EcsBatchConfig(
9         conf.ClusterName,
10        conf.TaskDefinition,
11        ... // omitted for brevity
12        new List<string> { CommandName, CommandParamBatchTypeId, detail.BatchTypeId.ToString() }
13    );
14    config.EnvironmentVariables.Add(EnvVarTraceId, traceId); // For monitoring batch
15    if (detail.Parameters?.Any() ?? false)
16        // expose event parameters to batch
17        config.EnvironmentVariables.Add(EnvVarBatchParameters, detail.Parameters);
18    return config;
19 }
```



```
1  execute_ecs_tasks = {  
2    effect = "Allow"  
3    actions = [  
4      "ecs:RunTask",  
5      "iam:PassRole"  
6    ]  
7    resources = ["batch-apps-*"]  
8    principals = []  
9  }  
10
```



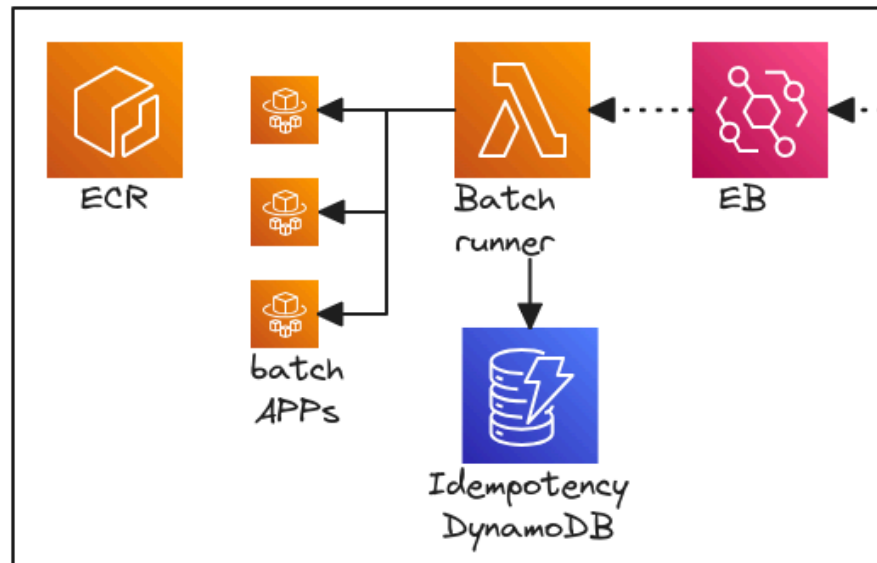
# wait

What about duplicated  
events?

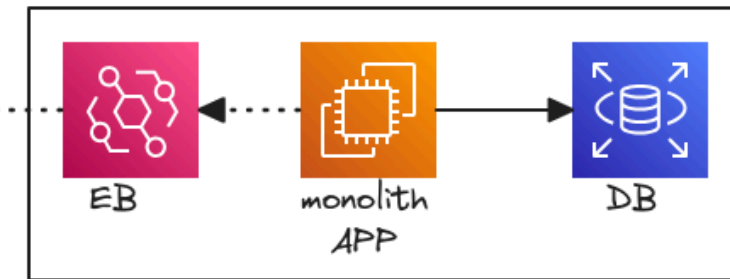


# Idempotency with DynamoDB

*New AWS account*

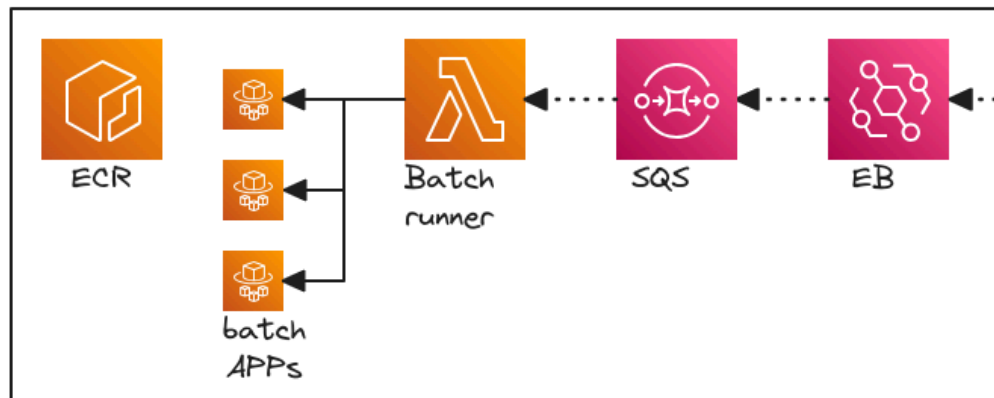


*Legacy AWS account*

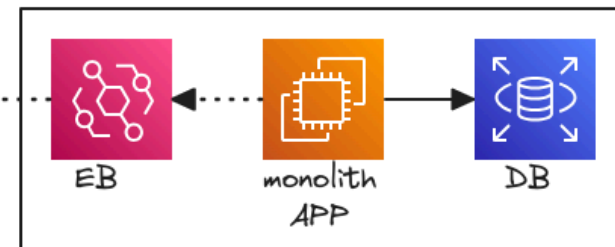


# Idempotency with SQS

*New AWS account*

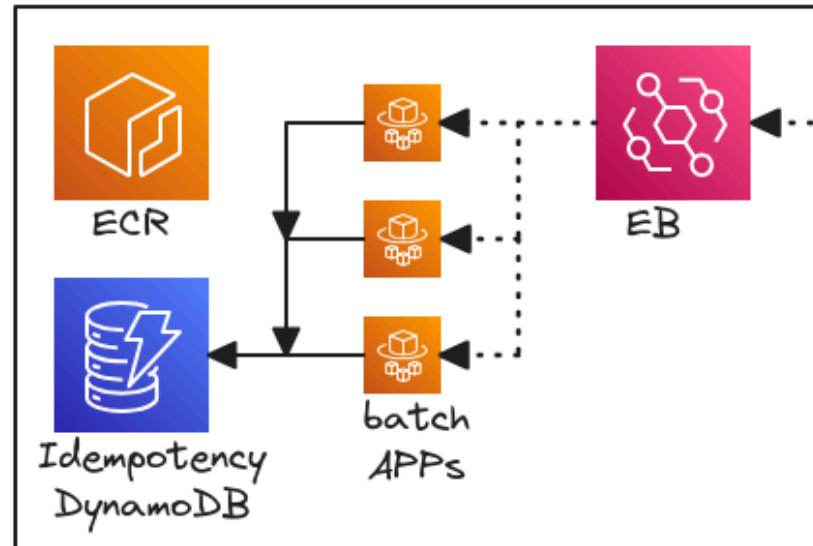


*Legacy AWS account*

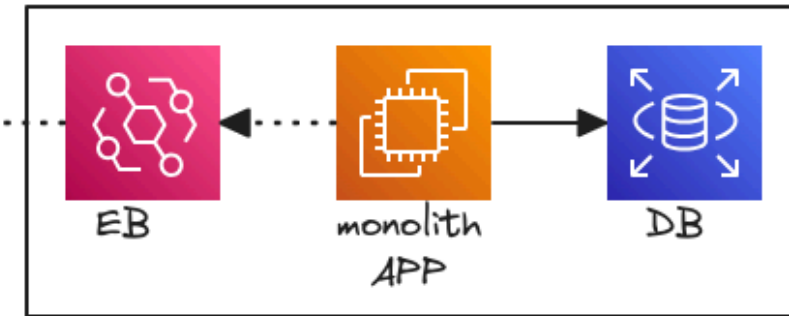


**simpler  
solution?**

*New AWS account*



*Legacy AWS account*



# What we learn

- lambda (+ SQS) offers more control
- EB + ECS is simpler but more expensive and adds logic to batch apps

# Questions?